**openssl生成证书，并配置HTTPS服务器**

**安装openssl**

yum install openssl -y

生成配置文件：/etc/pki/tls/openssl.cnf

文件中值得注意的配置项：

新证书存放位置

new\_certs\_dir = $dir/newcerts

CA私钥

private\_key = $dir/private/cakey.pem

证书索引数据库文件

database = $dir/index.txt

颁发证书的序列号

serial = $dir/serial

**创建CA私钥**

参数说明：

genras 使用 rsa 算法生成密钥。

-des3 （可选）加密密钥，此时需要设置密码，后续使用该密钥时需要验证密码才能使用。

-out 生成私钥文件。

cd /etc/pki

openssl genrsa -des3 -out ca.key 2048



输入密码随意例如：123456

**生成证书请求文件(CSR)**

参数说明：

req 产生证书签发申请命令。

-new 新的申请。

-key 输入的 key 文件，由第一步生成。

-out 输出为 CSR 文件，这是一个请求文件。

openssl req -new -key ca.key -out ca.csr

运行此命令后进入交互模式，需要输入一些证书信息。

先验证密码。

然后一般需要输入的信息如下：

C 国家

ST 省份

L 市

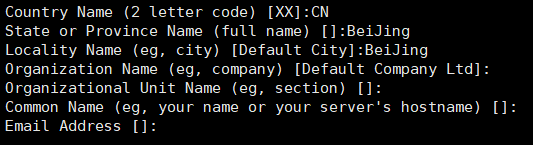
O 机构

OU 部门

CN (Common Name) 一般是域名

emailAddress 邮箱

例如如下（后面一堆我直接回车跳过）：



**自签署证书**

正常的证书是你把上面生成的请求文件(.CSR)发送给可信机构(CA)，让可信机构根据你的请求去生成和签署证书，再给你发回来。这里是自己给自己签署。需验证密码。

openssl x509 -req -sha256 -days 3650 -in ca.csr -signkey ca.key -out ca.crt

参数说明：

x509 签发X.509格式证书命令。

-req 证书输入请求。

-days 证书有效天数。

-in 输入文件，这里是上一步生成的请求文件(.CSR)

-signkey 签名密钥(key)文件，由第一步生成。

-out 输出文件，生成证书文件(.CRT)。

到这里，公钥就生成了。

ca.crt

ps:以上生成证书请求文件和自签名证书两步可以合成一步来执行：

openssl req -new -x509 -sha256 -days 3650 -key ca.key -out ca.crt

一般情况下，上面的 key 和 crt 可以直接拿来应用了。  
以下演示把当前的证书当成 CA 给其他的请求进行颁发证书。

**创建目录**

mkdir /etc/pki/CA/server

cd /etc/pki/CA/server/

**创建服务器私钥**

openssl genrsa -out server.key 2048

参数说明：

genras 使用 rsa 算法生成密钥。

-out 生成私钥文件。

**生成证书请求文件(CSR)**

openssl req -new -key server.key -out server.csr

**使用 CA 证书进行签署，生成 crt 文件**

输入服务器给的 csr 请求文件，使用指定 CA 的私钥和证书来签署，输出服务器证书 crt

openssl x509 -req -sha256 -days 3650 -in server.csr -CA /etc/pki/ca.crt -CAkey /etc/pki/ca.key -CAcreateserial -out server.crt

输入密码后生成crt证书

**实践总结**

CA 证书和服务器证书的区别只有最后签署时，是自己给自己签署，还是让别人给你签署。

实际应用时需要的是私钥(key)和证书(crt)文件。其中私钥文件很重要，不要公开出去。证书文件可以随意分发。

把 crt 证书文件加入可信任的根机构中，则该证书和其签署的所有证书都会被信任。那么在一个机构内部可以自建一个 CA 证书，CA 证书加入可信列表，然后机构内部的所有其他证书都使用该证书来签署，则只需要信任一次就够了。

**HTTPS的配置：**

**apache服务器配置：**

确认ssl模块（mod\_ssl.so）开启

编辑httpd.conf

LoadModule ssl\_module libexec/apache2/mod\_ssl.so

Include conf/extra/httpd-ssl.conf

Include conf/extra/httpd-vhosts.conf

编辑conf/extra/httpd-ssl.conf文件

SSLCertificateFile "/usr/local/server/apache/conf/server.crt"

SSLCertificateKeyFile "/usr/local/server/apache/conf/server.key"

设置https相应的虚拟端口配置，默认为443端口

编辑conf/extra/httpd-vhosts.conf文件

<VirtualHost \*:443>

ServerName www.example.com

DocumentRoot /www/example.com/htdocs

SSLengine on

SSLProtocol all -SSLv3

SSLcertificatefile /usr/local/server/apache/conf/server.crt

SSLcertificatekeyfile /usr/local/server/apache/conf/server.key

</VirtualHost>

配置SSL  
1.编辑 conf/extra/httpd-ssl.conf 文件  
httpd-ssl.conf 中已经有一条 <VirtualHost> 记录，我们将其注释掉，新建一条：

<VirtualHost \*:443>

重启httpd即可

**nginx配置**

源码下载地址：

wget https://www.openssl.org/source/openssl-1.0.2n.tar.gz

tar zxf openssl-1.0.2n.tar.gz

mv openssl-1.0.2n /opt/

编译nginx前配置，让nginx支持ssl\_module与openssl

./configure --prefix=/usr/local/nginx --with-http\_ssl\_module --with-openssl=/opt/openssl-1.0.2n

make && make install

编辑文件：

vi /usr/local/nginx/conf/nginx.conf

主要配置：

server {

listen 443;

ssl on;

#server\_name host.httpsDomain.com; #申请证书的域名

ssl\_certificate /etc/pki/CA/server/server.crt;

ssl\_certificate\_key /etc/pki/CA/server/server.key;

ssl\_session\_timeout 5m;

ssl\_protocols TLSv1 TLSv1.1 TLSv1.2 SSLv2 SSLv3; #指定SSL服务器端支持的协议版本

ssl\_ciphers ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP; #指定加密算法

ssl\_prefer\_server\_ciphers on; #在使用SSLv3和TLS协议时指定服务器的加密算法要优先于客户端的加密算法

启动：

/usr/local/nginx/sbin/nginx

**对于 IIS 的设置**

在 IIS 中，需要的是一个 PFX 文件，这个文件需要包含 key 和 crt。生成方法如下：

# 输入 key 和 crt 文件，输出 pfx 文件。

openssl pkcs12 -export -inkey server.key -in server.crt -out server.pfx

执行上述命令时，会要求输入一个 export 密码。 该密码在导入 pfx 文件时需要。

在 IIS 中选择“导入证书”，文件选择该 pfx 文件，密码填写导出时的密码，导入位置选择“个人”。

如要修改证书的 friendly name ，则在证书管理中修改（需要从 mmc 中打开计算机级别的证书管理器）。

**解决 Chrome 报 missing\_subjectAltName 的问题**

chrome 会查看当前域名是否在证书中声明，该声明由 subjectAltName 字段设置。上述的生成步骤默认未设置该字段。

解决方法如下：

新建一个文件，起名为 v3.ext (名字自定)，编辑内容如下：

subjectAltName = @alt\_names

[alt\_names]

DNS.1 = www.company.com

DNS.2 = company.com

DNS.3 = \*.company.net

域名要与你的证书实际绑定的域名一致。如有多个域名，按示例写多个。

在签署时，额外增加一个参数： -extfile v3.ext

# 这是上面服务器签署的示例，只在最后增加一个参数，指定扩展字段的配置文件。

openssl x509 -req -sha256 -days 3650 -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.cr

**通过配置文件，非交互式生成证书请求文件(CSR)**

新建一个文件，如 my.cnf ，相应字段按自己需求修改：

[req]

default\_bits = 2048

prompt = no

default\_md = sha256

distinguished\_name = req\_distinguished\_name

[req\_distinguished\_name]

C = CN

ST = GuangDong

L = ShenZhen

O = your\_company

OU = your\_organize

CN = www.myserver.com

emailAddress = admin@myserver.com

在生成请求文件时，额外增加参数： \*\*-config my.cnf \*\*

指定配置文件，此时不会出现交互模式，相交信息自动设置。

openssl req -new -key server.key -out server.csr -config my.cnf

**一句话生成 key 和 crt 文件**

该情形适合于自签署证书时。

请求的配置和 ext 的配置可以写在一起，如下

[req]

default\_bits = 2048

prompt = no

default\_md = sha256

distinguished\_name = req\_distinguished\_name

x509\_extensions = v3\_req

[req\_distinguished\_name]

C = CN

ST = GuangDong

L = ShenZhen

O = your\_company

OU = your\_organize

CN = www.myserver.com

emailAddress = admin@myserver.com

[v3\_req]

keyUsage = critical, digitalSignature, keyAgreement

extendedKeyUsage = serverAuth

subjectAltName = @alt\_names

[alt\_names]

DNS.1 = myserver.com

DNS.2 = \*.myserver.com

然后运行如下命令：

一句命令，输入配置文件 my.conf， 输出 key 和 crt 文件。适合于自签署证书。

openssl req -x509 -sha256 -nodes -days 3650 -newkey rsa:1024 -keyout app.key -out app.crt -config my.conf

**自制脚本**

参见： <https://github.com/ljskr/ssl_tool>